A97-32417 AIAA-97-2100

A NEW IMPLICIT ALGORITHM FOR FLUID FLOW

Robert W. MacCormack*
Department of Aeronautics and Astronautics
Stanford University
Stanford, CA 94305-4035

Abstract

An implicit procedure for minimizing the error induced by approximate factorization is presented for solving the unsteady Euler or Navier-Stokes equations. Convergenge rates of approximately 0.8 are achieved upon application for high Reynold's number supersonic flow past a blunt body and transonic and subsonic flow through a nozzle. Converged solutions, to at least engineering accuracy, are obtained in about 50 time steps. The operation count is about two and a half times that for a standard approximately factored procedure per time step. However, standard approximately factored procedures are severely limited by the error induced by approximate factorization and often require a thousands of time steps for convergence.

Introduction

Implicit procedures for solving the equations of compressible flow have been developed and used widely during the past quarter century. There is some thought that their development has reached maturity and that the field is now fairly barren for further development, at least with regard to application on structured grids and on computers with non-parallel architectures. However, this paper intends to show that there is some point to gleaning the field further, at least, first, for unifying our understanding of implicit procedures and second, for the development of highly efficient procedures for the powerful computer workstations availible today using conventional architectures. These are the two main goals of this paper.

Present implicit methods for solving the equations of fluid flow appoximate the set of governing partial differential equations with a set difference equations in matrix form. The mesh upon which the flow is descretized is assumed for this paper to be structured and two or three dimensional. The matrix equations are then, at present, usually solved by one of the following procedures.

1 Approximate L-U Decomposition via the Strongly Implicit Procedure (SIP)
2 Approximate Factorization (AF) for block tridiagonal Inversion
3 Gauss-Seidel Line Relaxation (GSLR)
4 Direct Inversion via Gaussian Elimination

The SIP procedure was presented by Stone[1] in 1968 and applied to solve the heat equation in two and three dimensions. In two dimensions the heat equation or the equations governing fluid flow are represented by a matrix equation containing five diagonals, not all adjacent. This matrix equation has no direct efficient inversion procedure. Stone approximated this matrix by the product of two matrices, one lower and the other upper bidiagonal, each with efficient inversion procedures. The product of the two matrices returned the original non-zero matrix elements, but also contributed new elements as well. It also required significantly increased storage memory for intermediate results during the inversion procedure, prohibitively so in three dimensions.

Approximate factorization has been the work horse of implicit procedures for fluid flow solutions during the past two decades. This procedure has been popularized by the Beam-Warming[2] and Briley-McDonald[3] methods. As in the SIP procedure for two dimensions, it approximates the original matrix by the product of two easily invertible matrices. The AF matrices are each tridiagonal. However, upon multiplication of the factors none of the orginal elements are returned in general, which causes slow convergence. But, this procedure requires very little memory for intermediate results because it inverts each matrix factor a (grid) line at a time.

Gauss-Seidel line relaxation has been applied to solve the Euler[4] and Navier-Stokes[5,6] equations during the past decade. It also uses tridiagonal inversion to solve for the unknowns simultaneously along a grid line and includes the latest values of the off line terms in Gauss-Seidel fashion. It is unfactored, but introduces a preferred direction by the choice of the line direction, usually across the finest distribution of mesh points where implicitness is most important. It therefore suffers when the choice is not clear, for example, for flow with corners and intersecting thin boundary layers.

The fourth procedure represents a true Newton procedure and usually requires the use first of a procedure of type 1,2, or 3 listed above to get close enough to the converged solution, for non-linear equations, to be effective. It is computationally expensive, prohibitively

* Professor, Fellow AIAA

so in three dimensions at present. Yet, this method, through the use of Krylov subspaces, has very high potential and ranks with the multi-grid procedure as a contender for the future preferred method for solving the equations of fluid flow. Both need further development to be competitive and to dominate the other implicit procedures listed above.

The first two procedures, SIP and AF, are approximate matrix decomposition procedures. Some would call them matrix preconditioning procedures as well. The third, GSLR, could also be classified this way, but remains distinct in that it is also an iterative procedure between time steps. The other two are said to be direct, though, unlike the fourth procedure, with approximate matrices.

In 1987 Bardina and Lombard[7] presented a diagonally dominant approximately factored procedure DDADI (Diagonally Dominant Alternating Direction Implicit) used within their upwind 3-D CSCM method (Conservative-Supra-Characteristic method) for solving the Navier-Stokes equations. Apparently independently, MacCormack[8] devised this same decomposition by transforming the SIP L-U decomposition into an approximately factored procedure containing tridiagonal matrices. This procedure retains the advantages of each and the main disadvantages of neither. It is clearly superior to SIP and conventional AF because:

(1) The procedure upon multiplication of the matrix factors returns exactly the non-zero elements of the original matrix, and

(2) The procedure solves the implicit matrix equation a line at a time, requiring little additional storage even in three dimensions.

However, like both SIP and conventional AF, upon multiplication of the approximating tridiagonal matrix factors, this modified approximate factorization (MAF) procedure introduces new matrix elements in positions that are zeros in the original matrix. These elements are the errors of approximate decomposition and represent sources and sinks of solution mass, momentum and energy changes, which will slow convergence, though not nearly so severely as conventional AF.

Finally, the paper presents an efficient iterative procedure for removing the errors of approximate factorization. These elements, MAF and the procedure for minimizing the error induced by MAF, are combined to give an efficient, believed to be new, procedure for solving the Navier-Stokes equations. This procedure is then applied to solve for high Reynold's number supersonic flow past a blunt body and transonic and subsonic flow through a nozzle.

Personal Note: The title of this paper includes the word *NEW*. It is risky and perhaps foolish to use this word in the already well gleaned field of implicit matrix algorithms. Each time I have presented research on this subject recently, someone always points out similar or identical work presented earlier. First, as an invited speaker at the Ohio Aerospace Institute, March 1996, the similarities of the decompositions I presented with Stone's SIP procedure were strongly noted. Second, on presentation at the ICNMFD meeting in Monterey, CA, June 1996, the earlier work of Bardina and Lombard[7] was pointed out to me. And, finally after a seminar I gave at NASA Ames in October 1996, I received in the mail from the Director of NASA Ames, H. McDonald a report[9] of his showing a similar procedure for error minimization. I am beginning to believe that either this subject is snake bitten or I have descended early into incompetency. Perhaps, I am at best a rediscoverer or a collector of useful procedures, some previously abandoned, that can be combined to produce a *new* method for solving the equations of fluid dynamics. Nevertheless, even if turns out not to be a *new* procedure, I believe that it is superior to those in wide use today and I will be satisfied to be its champion.

Implicit Matrix Equations

Consider the unsteady equations of compressible viscous flow in two dimensions represented in general coordinates ξ and η as follows.

$$\frac{\partial U}{\partial t} + \frac{\partial F}{\partial \xi} + \frac{\partial G}{\partial \eta} = 0 \tag{1}$$

A finite difference/volume equation, written in *delta* law form, approximating the above equation, is given by Eq. (2) below (2)

$$\left\{ I + \Delta t \left(\frac{D_{-}\cdot}{\Delta \xi} \bar{A}^n{}_{i+1/2,j} + \frac{D_{-}\cdot}{\Delta \eta} \bar{B}^n{}_{i,j+1/2} \right) \right\} \delta U^{n+1}_{i,j}$$
$$= -\Delta t \left(\frac{F^n_{i+1/2,j} - F^n_{i-1/2,j}}{\Delta \xi} + \frac{G^n_{i,j+1/2} - G^n_{i,j-1/2}}{\Delta \eta} \right)$$

where $\bar{A}$ and $\bar{B}$ are the matrix jacobians of the flux vectors F and G with respect to state vector U, respectively, and the dots, $\cdot$, appearing in the equation indicate that the difference operators operate on all factors to the right. Upon application of the difference operators an equation of the following form is usually obtained.

$$\mathbf{B_{i,j}} \delta U^{n+1}_{i,j+1} + \mathbf{A_{i,j}} \delta U^{n+1}_{i,j} + \mathbf{C_{i,j}} \delta U^{n+1}_{i,j-1} + \mathbf{D_{i,j}} \delta U^{n+1}_{i+1,j} + \mathbf{E_{i,j}} \delta U^{n+1}_{i-1,j} = \Delta U^n_{i,j} \tag{3}$$

where $\mathbf{A_{i,j}}$, $\mathbf{B_{i,j}}$, $\mathbf{C_{i,j}}$, $\mathbf{D_{i,j}}$ and $\mathbf{E_{i,j}}$ are block matrix elements and

$$\Delta U^n_{i,j} = -\Delta t\left(\frac{F^n_{i+1/2,j} - F^n_{i-1/2,j}}{\Delta\xi} + \frac{G^n_{i,j+1/2} - G^n_{i,j-1/2}}{\Delta\eta}\right)$$

Eq. (3) can also be expressed in full matrix form.

$$\mathbf{M}\cdot[\delta\mathbf{U}] = [\mathbf{\Delta U}]$$

where $\mathbf{M} =$

$$\begin{bmatrix}
x & x & \cdot & \cdot & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
x & x & x & \cdot & \cdot & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & x & x & x & \cdot & \cdot & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & x & x & x & \cdot & \cdot & x & \cdot & \cdot & \cdot & \cdot & \cdot \\
x & \cdot & \cdot & x & x & x & \cdot & \cdot & x & \cdot & \cdot & \cdot & \cdot \\
\cdot & x & \cdot & \cdot & x & x & x & \cdot & \cdot & x & \cdot & \cdot & \cdot \\
\cdot & \cdot & \mathbf{D} & \cdot & \cdot & \mathbf{B} & \mathbf{A} & \mathbf{C} & \cdot & \cdot & \mathbf{E} & \cdot & \cdot \\
\cdot & \cdot & \cdot & x & \cdot & \cdot & x & x & x & \cdot & \cdot & x & \cdot \\
\cdot & \cdot & \cdot & \cdot & x & \cdot & \cdot & x & x & x & \cdot & \cdot & x \\
\cdot & \cdot & \cdot & \cdot & \cdot & x & \cdot & \cdot & x & x & x & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & \cdot & \cdot & x & x & x & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & \cdot & \cdot & x & x & x \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & \cdot & \cdot & x & x
\end{bmatrix},$$

$$[\delta\mathbf{U}] = \begin{bmatrix} \cdot \\ \cdot \\ \delta U_{i-1,j} \\ \cdot \\ \cdot \\ \delta U_{i,j-1} \\ \delta U_{i,j} \\ \delta U_{i,j+1} \\ \cdot \\ \cdot \\ \delta U_{i+1,j} \\ \cdot \\ \cdot \end{bmatrix} \quad \text{and} \quad [\mathbf{\Delta U}] = \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \Delta U_{i,j} \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix}$$

Matrix $\mathbf{M}$ is of size $[(nXn)X(nXn)]$ and vectors $[\delta\mathbf{U}]$ and $[\mathbf{\Delta U}]$ are of length (nXn) on an (nXn) mesh.

The matrix $\mathbf{M}$ can be approximately factored into two block tridiagonal factors $\mathbf{T}_\xi$ and $\mathbf{T}'_\eta$.

$$\mathbf{M} \simeq \mathbf{T}_\xi \cdot \mathbf{T}'_\eta$$

where $\mathbf{T}_\xi =$

$$\begin{bmatrix}
x & \cdot & \cdot & \cdot & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & x & \cdot & \cdot & \cdot & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & x & \cdot & \cdot & \cdot & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & x & \cdot & \cdot & \cdot & x & \cdot & \cdot & \cdot & \cdot & \cdot \\
x & \cdot & \cdot & \cdot & x & \cdot & \cdot & \cdot & x & \cdot & \cdot & \cdot & \cdot \\
\cdot & x & \cdot & \cdot & \cdot & x & \cdot & \cdot & \cdot & x & \cdot & \cdot & \cdot \\
\cdot & \cdot & \mathbf{D} & \cdot & \cdot & \cdot & \mathbf{A} & \cdot & \cdot & \cdot & \mathbf{E} & \cdot & \cdot \\
\cdot & \cdot & \cdot & x & \cdot & \cdot & \cdot & x & \cdot & \cdot & \cdot & x & \cdot \\
\cdot & \cdot & \cdot & \cdot & x & \cdot & \cdot & \cdot & x & \cdot & \cdot & \cdot & x \\
\cdot & \cdot & \cdot & \cdot & \cdot & x & \cdot & \cdot & \cdot & x & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & \cdot & \cdot & \cdot & x & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & \cdot & \cdot & \cdot & x & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & \cdot & \cdot & \cdot & x
\end{bmatrix}$$

and $\mathbf{T}'_\eta =$

$$\begin{bmatrix}
\mathbf{I} & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
x & \mathbf{I} & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & x & \mathbf{I} & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & x & \mathbf{I} & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & x & \mathbf{I} & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & x & \mathbf{I} & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \mathbf{B'} & \mathbf{I} & \mathbf{C'} & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & \mathbf{I} & x & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & \mathbf{I} & x & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & \mathbf{I} & x & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & \mathbf{I} & x & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & \mathbf{I} & x \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & \mathbf{I}
\end{bmatrix}$$

In the above matrix definitions, the subscripts on block matrix elements $\mathbf{A_{i,j}}$, $\mathbf{B_{i,j}}$, $\mathbf{C_{i,j}}$, $\mathbf{D_{i,j}}$ and $\mathbf{E_{i,j}}$ have been left off for convenience and the other non-zero elements were represented merely by x. Also, in the above matrix the primed elements are defined by $\mathbf{B'_{i,j}} = \mathbf{A^{-1}_{i,j}B_{i,j}}$ and $\mathbf{C'_{i,j}} = \mathbf{A^{-1}_{i,j}C_{i,j}}$.

Alternatively, the decomposition for matrix $\mathbf{M}$ could be written as

$$\mathbf{M} \simeq \mathbf{T}_\xi \cdot [\mathbf{A}]^{-1} \cdot \mathbf{T}_\eta$$

where $[\mathbf{A}]^{-1}$ is a diagonal matrix given by

$$\begin{bmatrix}
x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \mathbf{A^{-1}_{i,j}} & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x
\end{bmatrix}$$

and $\mathbf{T}_\eta =$

$$\begin{bmatrix}
x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
x & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & x & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & x & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & x & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & x & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \mathbf{B} & \mathbf{A} & \mathbf{C} & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & x & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & x & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & x & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & x & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & x \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x
\end{bmatrix}$$

In three dimensions $\mathbf{M}$ can be decompositioned[7,8] by

$$\mathbf{M} \simeq \mathbf{T}_\xi \cdot [\mathbf{A}]^{-1} \cdot \mathbf{T}_\eta \cdot [\mathbf{A}]^{-1} \cdot \mathbf{T}_\zeta$$

Decomposition Error

The original matrix **M** is returned upon multiplication of the matrix factors, but in addition some previously zero elements are modified.

$$\mathbf{M} \simeq \mathbf{T}_\xi \cdot \mathbf{T}'_\eta = \mathbf{M} + \mathbf{P}$$

where $\mathbf{M} + \mathbf{P} =$

$$\begin{bmatrix}
x & x & \cdot & x & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
x & x & x & \cdot & x & x & x & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & x & x & x & \cdot & x & x & x & \cdot & \cdot & \cdot & \cdot & \cdot \\
x & \cdot & x & x & x & \cdot & x & x & x & \cdot & \cdot & \cdot & \cdot \\
x & x & \cdot & x & x & x & \cdot & x & x & x & \cdot & \cdot & \cdot \\
x & x & x & \cdot & x & x & x & \cdot & x & x & x & \cdot & \cdot \\
\cdot & \mathbf{F} & \mathbf{D} & \mathbf{G} & \cdot & \mathbf{B} & \mathbf{A} & \mathbf{C} & \cdot & \mathbf{R} & \mathbf{E} & \mathbf{S} & \cdot \\
\cdot & \cdot & x & x & x & \cdot & x & x & x & \cdot & x & x & x \\
\cdot & \cdot & \cdot & x & x & x & \cdot & x & x & x & \cdot & x & x \\
\cdot & \cdot & \cdot & \cdot & x & x & x & \cdot & x & x & x & \cdot & x \\
\cdot & \cdot & \cdot & \cdot & \cdot & x & x & x & \cdot & x & x & x & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & x & \cdot & x & x & x \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & x & x & x & \cdot & x & x
\end{bmatrix}$$

The block matrix elements introduced by approximate decompostion, shown above, are $\mathbf{F_{i,j}}$, $\mathbf{G_{i,j}}$, $\mathbf{R_{i,j}}$ and $\mathbf{S_{i,j}}$, where

$$\mathbf{F_{i,j}} = \mathbf{D_{i,j}}\mathbf{B'_{i-1,j}}$$
$$\mathbf{G_{i,j}} = \mathbf{D_{i,j}}\mathbf{C'_{i-1,j}}$$
$$\mathbf{R_{i,j}} = \mathbf{E_{i,j}}\mathbf{B'_{i+1,j}}$$
$$\mathbf{S_{i,j}} = \mathbf{E_{i,j}}\mathbf{C'_{i+1,j}}$$

Instead of Eq. (3), the equation actually solved is

$$\begin{aligned}
&\mathbf{B_{i,j}}\delta U^{n+1}_{i,j+1} + \mathbf{A_{i,j}}\delta U^{n+1}_{i,j} + \mathbf{C_{i,j}}\delta U^{n+1}_{i,j-1} \\
&\quad + \mathbf{D_{i,j}}\delta U^{n+1}_{i+1,j} + \mathbf{E_{i,j}}\delta U^{n+1}_{i-1,j} \\
&\quad + \mathbf{F_{i,j}}\delta U^{n+1}_{i+1,j+1} + \mathbf{G_{i,j}}\delta U^{n+1}_{i+1,j-1} \\
&\quad + \mathbf{R_{i,j}}\delta U^{n+1}_{i-1,j+1} + \mathbf{S_{i,j}}\delta U^{n+1}_{i-1,j-1} = \Delta U^n_{i,j}
\end{aligned}$$

or

$$\mathbf{M} \cdot [\delta \mathbf{U}] + \mathbf{P} \cdot [\delta \mathbf{U}] = [\mathbf{\Delta U}] \qquad (4)$$

The term $\mathbf{P} \cdot [\delta \mathbf{U}]$ is the decomposition error. It represents erroneous sources and sinks for information carried by the matrix equation and can be expected to degrade convergence. Nevertheless, the above MAF procedure is stable, which will be shown later, and will converge faster than the usual AF procedures in use today within the popular Beam-Warming and Briley-McDonald methods.

Removal of Decomposition Error

We now present a procedure for removing decomposition error. The decomposition error term is fed back into the matrix equation on the right hand side for self cancellation. A k-step iterative algorithm, MAF(k), is defined as follows.

MAF(k) Algorithm

$$\mathbf{M} \cdot \left[\delta \mathbf{U^{(k)}}\right] + \mathbf{P} \cdot \left[\delta \mathbf{U^{(k)}}\right] = [\mathbf{\Delta U}] + \mathbf{P} \cdot \left[\delta \mathbf{U^{(k-1)}}\right]$$
$$\text{where} \quad \left[\delta \mathbf{U^{(0)}}\right] = [0] \qquad k = 1, 2, 3, \cdots$$

MAF(1) is the same as Eq. (4) above. For this iterative procedure to work, each MAF iteration must be numerically stable and the sequence must converge. A stability analysis will be presented in the next section and an example showing sequence convergence will be given in the section on numerical results.

Stability of MAF(k) Procedure

Consider the following model hyperbolic equation.

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} + b\frac{\partial u}{\partial y} = 0$$

with both a and $b > 0$. Using first order upwind difference approximations for all spacial derivatives and the MAF(1) algorithm described above, we obtain the following numerical amplification factor.

$$g = 1 - \frac{\hat{a}(1 - e^{-i\xi}) + \hat{b}(1 - e^{-i\eta})}{(1 - \hat{a}e^{-i\xi})(1 - \hat{b}e^{-i\eta})}$$
$$\text{where} \quad \hat{a} = a\frac{\Delta t}{\Delta x}/(1 + a\frac{\Delta t}{\Delta x} + b\frac{\Delta t}{\Delta y})$$
$$\text{and} \quad \hat{b} = b\frac{\Delta t}{\Delta y}/(1 + a\frac{\Delta t}{\Delta x} + b\frac{\Delta t}{\Delta y})$$

By rearrangement of terms, MAF(1) is shown below to be stable.

$$g = \frac{1 - \hat{a} - \hat{b} + \hat{a}\hat{b}e^{-i\xi}e^{-i\eta}}{(1 - \hat{a}e^{-i\xi})(1 - \hat{b}e^{-i\eta})}$$
$$\text{and} \quad |g| = \frac{(1 - \hat{a})(1 - \hat{b})}{(1 - \hat{a}e^{-i\xi})(1 - \hat{b}e^{-i\eta})} \leq 1$$

Similarly, the amplification factor $g^{(k)}$ for MAF(k), for all $k \geq 1$, is also stable, as shown below.

$$g^{(k)} = \frac{1 - \hat{a} - \hat{b} + \hat{a}\hat{b}e^{-i\xi}e^{-i\eta}g^{(k-1)}}{(1 - \hat{a}e^{-i\xi})(1 - \hat{b}e^{-i\eta})}$$
$$\text{with} \quad |g^{(k-1)}| \leq 1 \quad \text{and} \quad g^{(0)} = 1, \quad \text{and therefore}$$
$$\text{again} \quad |g^{(k)}| = \frac{(1 - \hat{a})(1 - \hat{b})}{(1 - \hat{a}e^{-i\xi})(1 - \hat{b}e^{-i\eta})} \leq 1$$

Numerical Method

A finite volume approach is used to descretize the equations on a structured mesh in general non-orthogonal coordinates. The inviscid spacial terms of Eq. (1) are approximated to third order accuracy using modified Steger-Warming flux vector splitting, which is shown below. The viscous terms are always central

differenced and are second order accurate. The difference operators appearing on the implicit side of Eq. (2) were first order accurate.

Inviscid Terms

For convenience of illustration, consider the flux terms of Eq. (1) as consisting only of the inviscid terms. The flux vector is approximated by

$$\begin{aligned} F_{i+1/2,j} &= \bar{A}_{+i+1/2,j} U_L + \bar{A}_{-i+1/2,j} U_R \\ U_L &= \frac{1}{8}(3U_{i+1,j} + 6U_{i,j} - U_{i-1,j}) \\ U_R &= \frac{1}{8}(3U_{i,j} + 6U_{i+1,j} - U_{i+2,j}) \\ \bar{U} &= \frac{1}{2}(U_L + U_R) \end{aligned}$$

and the flux split jacobians $\bar{A}_{+i+1/2,j}$ and $\bar{A}_{-i+1/2,j}$ are each functions of $\bar{U}$ only. The flux vector $G_{i,j+1/2}$ is similarly defined. The block element matrices are now defined as follows.

$$\mathbf{A_{i,j}} = I + \alpha\frac{\Delta t}{\Delta x}\left(\bar{A}^n_{+i+1/2,j} - \bar{A}^n_{-i-1/2,j}\right) + \alpha\frac{\Delta t}{\Delta y}\left(\bar{B}^n_{+i,j+1/2} - \bar{B}^n_{-i,j-1/2}\right)$$

$$\mathbf{B_{i,j}} = +\alpha\frac{\Delta t}{\Delta y}\bar{B}^n_{-i,j+1/2}, \qquad \mathbf{C_{i,j}} = -\alpha\frac{\Delta t}{\Delta y}\bar{B}^n_{+i,j-1/2},$$

$$\mathbf{D_{i,j}} = +\alpha\frac{\Delta t}{\Delta x}\bar{A}^n_{-i+1/2,j}, \qquad \mathbf{E_{i,j}} = -\alpha\frac{\Delta t}{\Delta x}\bar{A}^n_{+i-1/2,j}$$

and $\alpha \geq 1$ was chosen to be 2.

Viscous Terms

The full set of Navier-Stokes viscous terms were included in the right hand side term of Eq. (3), $\Delta U^n_{i,j}$, but only the pure ξ and η derivative terms were included in the left hand side block element matrix terms shown above. The mixed derivative terms were therefore neglected on the implicit side of the matrix equation. However, it can be shown that this is sufficient to maintain stability even on arbitrarily stretched grids.

Finally, the modified approximate factorization algorithm MAF(k) described above was then used to solve the implicit matrix equation at each time step.

Numerical Results

The numerical method just described was applied to solve the Navier-Stokes equations for supersonic flow past a sphere and transonic and supersonic flow through a converging-diverging nozzle.

The root-mean-square values for density residuals were calculated at each time step as follows.

$$Residual^n = \alpha_0 \sqrt{\frac{1}{I \cdot J} \sum_{i,j=1,1}^{I,J} \left(\frac{\Delta\rho^n_{i,j}}{\Delta t \cdot \rho^n_{i,j}}\right)^2}$$

where α_0 is a constant, $I \cdot J$ is the number of interior mesh points and $\Delta\rho^n_{i,j}$ is the continuity element of $\Delta U^n_{i,j}$.

The convergence rate was also assessed at each step, geometrically averaged over the previous m steps, as follows.

$$CR^n = \left(\frac{Residual^n}{Residual^{n-m}}\right)^{\frac{1}{m}}$$

and the value used for m was 10.

Supersonic Flow Past a Blunt Body

Mach 5 flow past a sphere of diameter 0.127m at a Reynolds number of $1.89\text{x}10^6$ was used for the first test case. The grid has a strong influence on the quality of the solution, particularly in the stagnation point region. The grid was highly stretched, with spacings as small as $7\text{x}10^{-7}$m and grid aspect ratios as high as $4\text{x}10^3$ near the sphere surface. Grid adaptation was used to obtain a good starting grid for this study. This grid, containing 42x50 finite volumes, was then held fixed for each algorithm studied and is shown in Fig. 1.

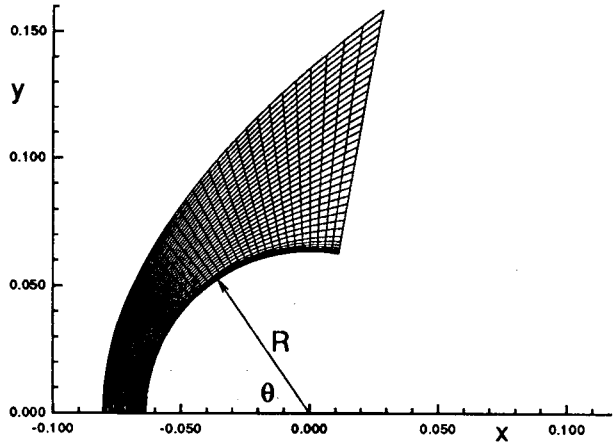

Figure 1. Computational mesh for supersonic flow past a sphere.

An initial bow shock wave location was chosen to be aligned with a grid line of the mesh. The Rankine-Hugoniot shock-jump relations were used to initialize the flow behind the shock wave with the normal-to-the-body-surface velocity component reducing to zero as the body surface was approached. The calculation started with a CFL number (ratio of time step used to the maximum allowed for an explicit method) of about 5000 and was increased to about $2.5\text{x}10^7$ over the next 50 time steps.

Computational results for pressure and Mach contours, using algorithm MAF(2), are shown in Figs. 2-3 below. Surface pressure, skin friction and heat transfer are shown in Fig. 4 with repect θ, in degrees measured from the stagnation point. Note that the solution is well behaved in the stagnation point region - skin friction increases linearly from zero until the boundary

begins to thicken and surface pressure and heat transfer are *bell* shaped curves with their maxima at the stagnation point.

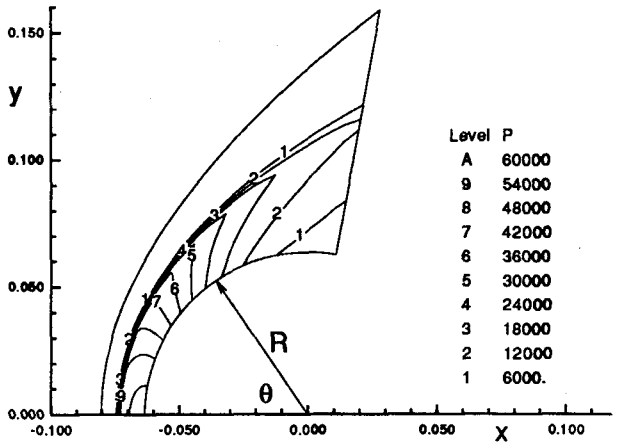

Figure 2. Pressure contours for Mach 5 flow past a sphere.

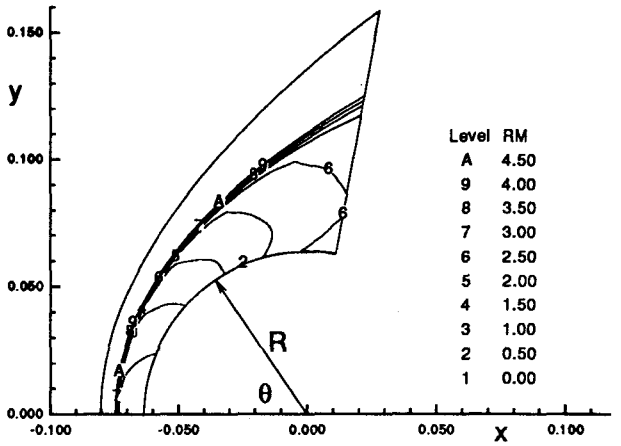

Figure 3. Mach contours for Mach 5 flow past a sphere.

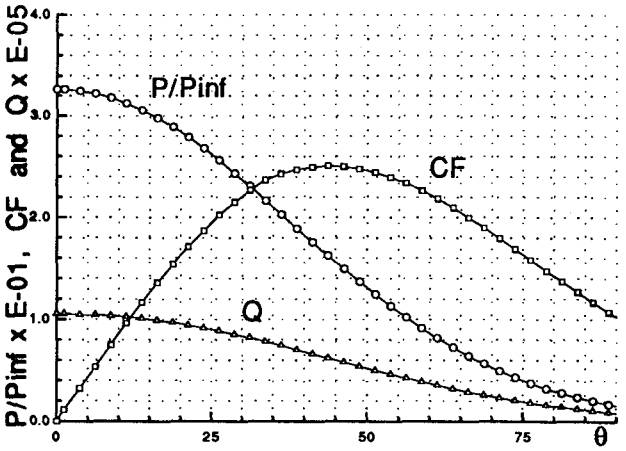

Figure 4. Surface pressure, skin friction and heat transfer for Mach 5 flow past a sphere (pressure is normalized by the freestream pressure, skin friction was devided by the dynamic freestream pressure and Q, the heat transfer, is given in Watts/m^2).

The residual reductions for algorithms MAF(1) through MAF(4) are compared with Gauss-Seidel line relaxation (GSLR) in Fig. 5. GSLR is exceptionally efficient for this class of flows containing a fine grid spanning only one coordinate direction.

Note that -

(a) the sequence of modified approximate factorization algorithms converges,

(b) all but MAF(1), which contains no error correction term, converges at about the same rate as GSLR and

(c) converged solutions are obtained in about 50 time steps.

The average convergence rate of MAF(2) from time step 21 through 55 is $CR = 0.78$. All methods, except MAF(1), are shown if Fig. 5 to reduce their residuals by 5 orders of magnitude, approximately to machine zero for the 32 bit word workstation used. Often, because of the switches used in flux-split methods, solutions reduce their residuals to only periodic limit cycles.

The operational count for algorithm MAF(k), for $k > 1$, is about $k + 0.5(k-1)$ times that for MAF(1) or conventional approximate factorization (AF). The $0.5(k-1)$ addition results from the evaluation of the error correction terms. Therefore, from the results shown in Fig. 5, algorithm MAF(2) appears to be the optimum choice. The cost of using MAF(2) is two and a half times as expensive as conventional AF. However, because of their diagonal dominance, the MAF algorithms can operate at exceptionally high CFL numbers, and should therefore be numerically more efficient than AF for solving the Navier-Stokes equations at high Reynolds numbers that require exceptionally fine grids.

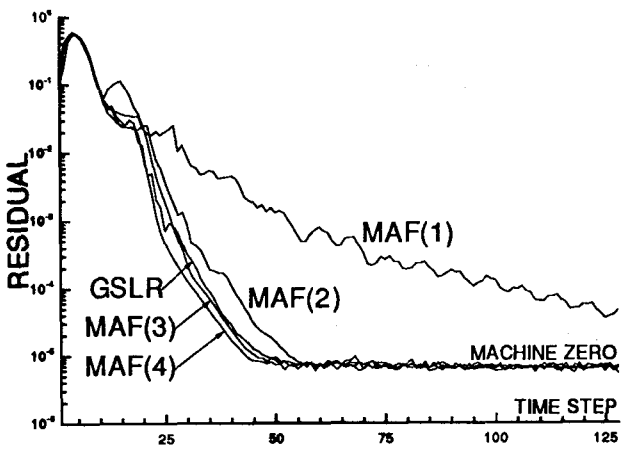

Figure 5. Residuals vs. time step for Mach 5 flow past a sphere.

Flow Through a Converging-Diverging Nozzle

Algorithm MAF(2) was applied to solve the Navier-

Stokes equations for flow through a converging-diverging nozzle. The 61x18 mesh used is shown in Fig. 6.

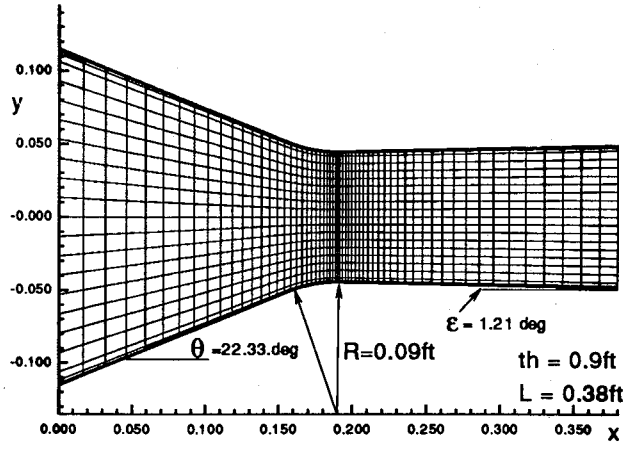

Figure 6. Computational mesh for converging-diverging nozzle

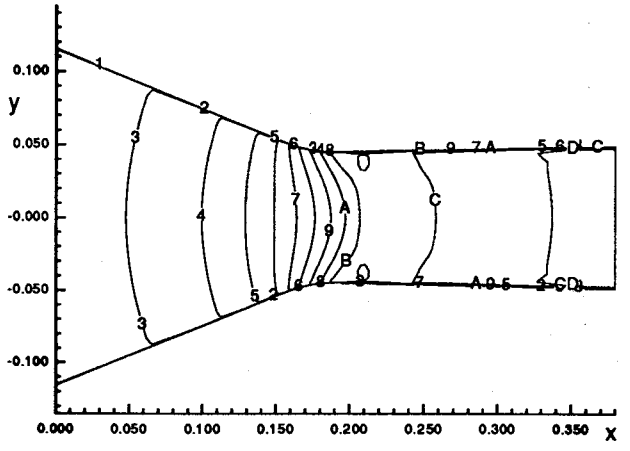

Figure 7. Mach contours for transonic flow through a converging-diverging nozzle. Contours are 0.1 apart and contour A is the sonic line.

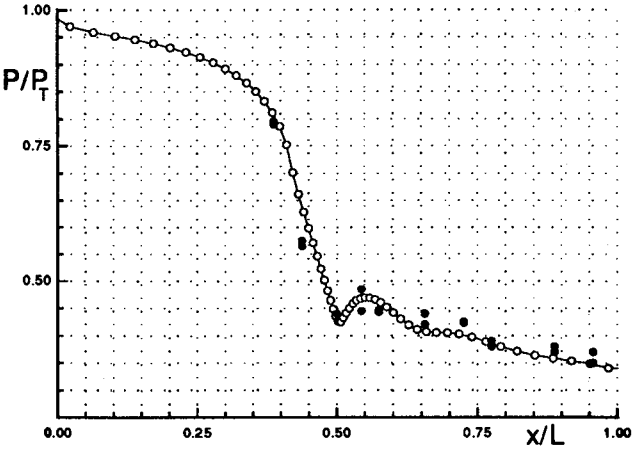

Figure 8. Comparison of numerical (open symbols) and experimental[10] results (solid symbols) for surface pressure for transonic flow through a converging-diverging nozzle

The calculation was started with the flow at rest at atmospheric conditions (2117.0 lb/ft^2 pressure) and a sudden reduction of pressure at the nozzle exit. A CFL number of approximately 70,000 was chosen initially and then was increased to a few million during the first 25 time steps. Three test cases, one transonic and two subsonic, with exit pressures of 600.0, 1800.0 and 2000.0 lb/ft^2, respectively, were run. Mach contours for the transonic case are shown in Fig. 7 and surface pressure results are compared with the experimental values of Mason[10] *et al* in Fig. 8. The Mach contours for the lowest subsonic case are shown in Fig. 9.

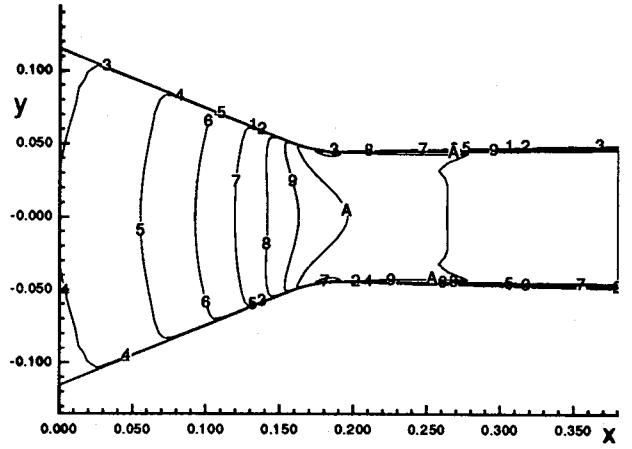

Figure 9. Mach contours for low subsonic flow through a converging-diverging nozzle. Contours are 0.03 apart and contour A defines Mach 0.3.

The reduction in residual with time step is shown for each case in Fig. 10. Note that again a converged solution for each case is obtained in about 50 time steps. The average convergence rate for the transonic case from time step 25 through time step 53 is 0.81. The other cases match this rate over the steep segments of their residual curves shown in Fig. 10. Each case reduces to a different residual level, depending apparently on how far the initial solution is from the equilibrium of steady state.

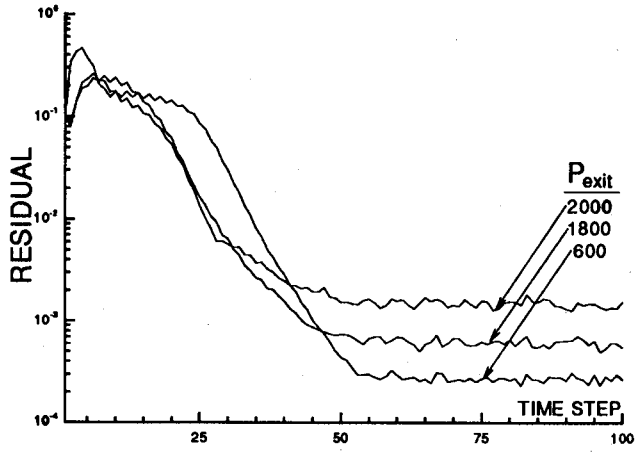

Figure 10. Residuals vs. time step for flow through a converging-diverging nozzle

Conclusion

A new class of general implicit algorithms, MAF(k), with high rates of convergence has been developed for solving the equations of compressible viscous flow on structured grids. High convergence rates are obtained because the algorithm minimizes the errors of approximate matrix decomposition in two ways. First, the decomposition preserves exactly the original nonzero matrix elements and second, the error introduced by approximate decomposition is fed back into the MAF(k) algorithm for self cancellation.

The algorithm was applied to high Reynolds number flow problems that contained shock waves, viscous effects of skin friction and heat transfer, and to low subsonic flow. Convergence rates of about 0.8 were achieved and steady state solutions were obtained in about 50 time steps in all cases presented. The choice of time step, whether for time dependent or steady state flow, is much less restricted than conventional approximately factored methods.

References

[1] Stone, H.L., "Iterative Solution of Implicit Approximations of Multidimensional Partial Differential Equations," *Siam J. Numer. Anal.*, Vol. 5, No. 3, Sept., 1968, pp. 530–558.

[2] Beam, R. and R.F. Warming, "An Implicit Factored Scheme for the Compressible Navier-Stokes Equations," *AIAA Journal*, Vol. 16, 1978, pp. 293–402.

[3] Briley, W.R. and H. McDonald, "Solution of the Multidimensional compressible Navier-Stokes Equations by a Generalized Implicit Method," *Journal of Computational Physics*, Vol. 24, 1977, pp. 372–397.

[4] Chakravarthy, S.R., "Relaxation Methods for Unfactored Upwind Schemes," *AIAA Paper No. 84–0165*, 1984.

[5] Thomas, J.L. and R.W. Walters, "Upwind Relaxation Algorithms for the Navier-Stokes equations," *AIAA Paper No. 85–1501*, 1985.

[6] MacCormack, R.W., "Current Status of Numerical Solutions of the Navier-Stokes equations," *AIAA Paper No. 85–0032*, 1985.

[7] Bardina, J. and C.K. Lombard, "Three Dimensional Hypersonic Flow Simulations with the CSCM Implicit Upwind Navier-Stokes Method," *AIAA Paper No. 87–1114*, 1987.

[8] MacCormack, R.W., "Efficient Matrix Decomposition for Implicit Algorithms," presented at *The 15th International Conference on Numerical Methods in Fluid Dynamics*, Monterey, CA June 24-28, 1996, proceedings to be published in *Lecture Notes in Phys., Spr.-Verlag*

[9] Briley, W.R., T.R. Govindan and H. McDonald, "Efficient Navier-Stokes Flow Prediction Algorithms," NASA Contract Final Report, Contract No. NAS8-37340, June 1990.

[10] Mason, M.L., L.E. Putnam and R.J. Re, "The Effect of Throat Contouring on Two-Dimensional Converging-Diverging Nozzles at Static Conditions," NASA TP-1704, 1980.